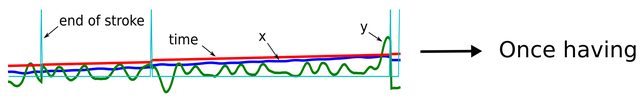
Unconstrained On-line Handwriting Recognition with Recurrent Neural Networks

Alex Graves[1], **Santiago Fernández**[2], **Marcus Liwicki**[3]
Horst Bunke[3], **Jürgen Schmidhuber**[1,2]

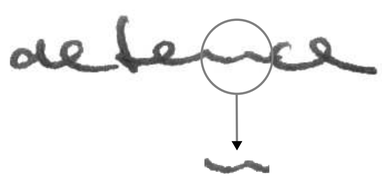
[1] TU Munich, Boltzmannstr. 3, 85748 Garching, Munich, Germany
[2] IDSIA, Galleria 2, 6928 Manno-Lugano, Switzerland
[3] IAM, University of Bern, Neubrckstrasse 10, CH-3012 Bern, Switzerland
{alex,santiago,juergen}@idsia.ch {liwicki,bunke}@iam.unibe.ch

On-line Handwriting Recognition

- On-line handwriting recognition extracts sequences of words from sequences of pen positions



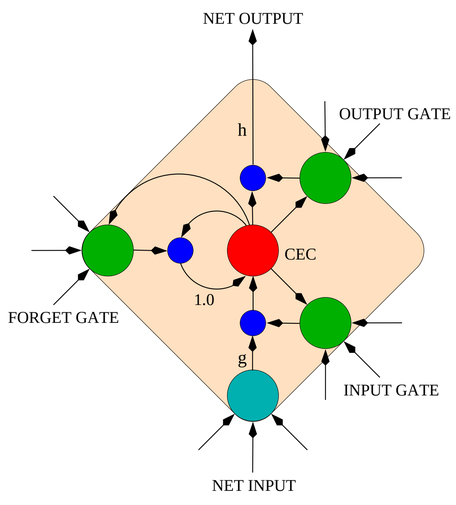
- It is generally easier than off-line recognition, where the text image is used as input
- However, each letter is spread over many pen positions and extensive use of context is required—a problem for most sequence learning algorithms



- Especially difficult for *unconstrained* handwriting, where any writing style can be used
- Usual solution is to pre-process the data into a set of localised features (containing information about shape, trajectory etc.)
- Special features are required for delayed strokes, e.g. the crossing of a 't' or the dot of an 'i'
- Would prefer to map directly from pen trajectories to transcriptions, since this (1) involves less human effort (2) is more general (e.g. could be used for languages with different alphabets) and (3) builds the feature extraction into the classifier, allowing the whole system to be trained together

Recurrent Neural Networks (RNNs)

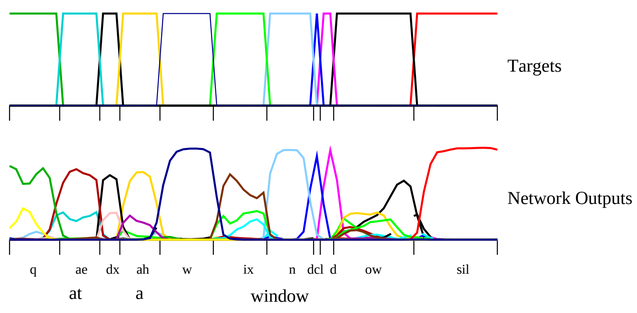
- RNNs provide a differentiable map from input to output sequences
- The recurrent connections give access to previous context and build in robustness to local distortions and translations of the input
- Bidirectional RNNs feed the input sequence forwards and backwards to two separate hidden layers, giving access to both past and future context (e.g. the letters before and after the current one)
- One problem with traditional RNNs is the limited range of context they can access, due to the so-called *vanishing gradient problem*
- The Long Short-Term Memory (LSTM) architecture uses a linear 'state' unit surrounded by multiplicative 'gates' to bridge long time delays, thereby giving access to long range context



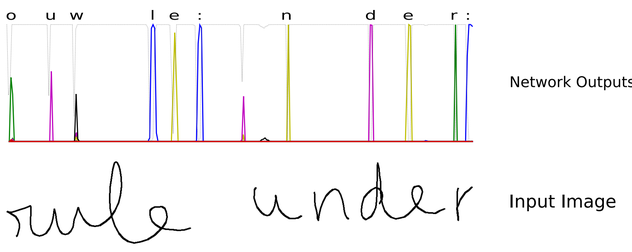
- Combining the above gives bidirectional LSTM (BLSTM)
- In principle BLSTM has access to context from the entire input sequence at every point in the output sequence

Connectionist Temporal Classification (CTC)

- The standard objective functions for RNNs require a separate training signal for each point in the input sequence



- This is a problem for tasks like handwriting recognition, where the alignment between the inputs and the labels is hard to determine automatically, and (at least for cursive text) the boundaries between letters are ambiguous
- CTC is a recently proposed output layer that trains RNNs to map directly from input sequences to label sequences, without requiring pre-alignment



- The labelling can be read directly from the network outputs (follow the spikes)
- The network can be trained with a log-likelihood objective function to maximise the probabilities of the correct labellings in the training set

Integration with an External Grammar

- CTC outputs the label sequence $\mathbf{l}^*$ with highest probability given the input sequence $\mathbf{x}$

$$\mathbf{l}^* = \arg\max_{\mathbf{l}} p(\mathbf{l}|\mathbf{x})$$

- But for certain tasks, we want to further weight the labellings according to some probabilistic grammar G (e.g. using bigram or trigram word probabilities)

$$\mathbf{l}^* = \arg\max_{\mathbf{l}} p(\mathbf{l}|\mathbf{x}, G).$$

- If we assume that $\mathbf{x}$ is independent of G, the above equation reduces to

$$\mathbf{l}^* = \arg\max_{\mathbf{l}} p(\mathbf{l}|\mathbf{x}) p(\mathbf{l}|G).$$

- This can be efficiently calculated with a variant of the *token passing algorithm* for HMMs

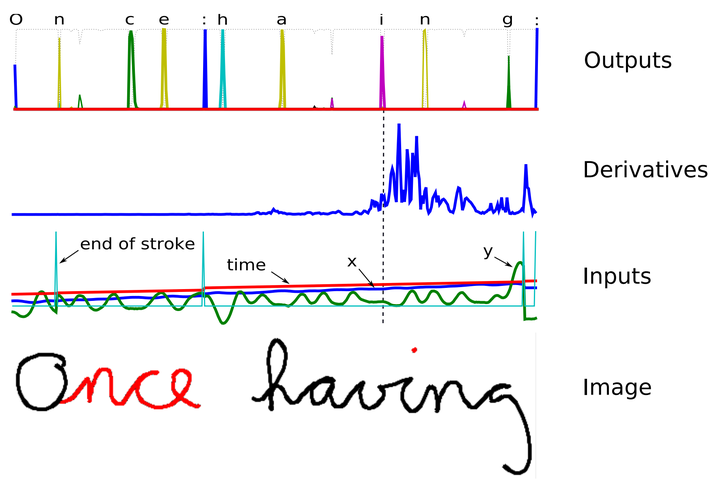
Experimental Data

- The IAM on-line database contains lines of unconstrained handwriting, with the pen positions captured from a "smart whiteboard" using an infrared receiver

- The input data consists of sequence of pen coordinates, along with the time the coordinate was recorded and an extra input to indicate when the pen was lifted off the board
- We used the inputs both raw and pre-processed with state-of-the-art techniques, such as skew and slant correction, size normalisation, feature extraction etc.
- A 20,000 word dictionary was used to constrain the output words, giving 94% coverage of the test set

Use of Context

- The main difference between the raw and pre-processed data was the amount of context required to identify letters (the pre-processing provides localised features that are easier to classify)
- The sensitivity of a CTC network to context can be analysed by plotting the *sequential Jacobian*—the derivative of the output at a particular point with respect to the inputs at all points in the sequence



- In this example, with raw inputs, the sequential Jacobian is plotted for the output corresponding to letter 'i' at the point when 'i' is emitted (shown by the dotted line)
- The range of sensitivity extends through most of the word 'having'
- Note the spike in sensitivity at the very end: this corresponds to the delayed dot of the 'i'

Results

- Identical BLSTM CTC networks were applied to the raw and pre-processed data
- The word error rate (WER) on the test set was calculated both with and without a bigram language model (LM)
- An HMM system was used for comparison

System	Input	LM	WER
HMM	pre-processed	✓	35.5%
CTC	raw	✗	30.1 ± 0.5%
CTC	pre-processed	✗	26.0 ± 0.3%
CTC	raw	✓	22.8 ± 0.2%
CTC	pre-processed	✓	20.4 ± 0.3%

- The CTC network substantially outperformed the HMM, with or without a language model, for both input types
- CTC performance was almost as good with raw inputs as with pre-processed
- It would be impossible to train most sequence learning algorithms (e.g. HMMs or traditional RNNs) with the raw inputs only